SmartQ: A Question and Answer System for Supplying High-Quality and Trustworthy Answers

Yuhua Lin, *Member, IEEE* and Haiying Shen, *Senior Member, IEEE*

Abstract—Question and Answer (Q&A) systems aggregate the collected intelligence of all users to provide satisfying answers for questions. A well-developed Q&A system should incorporate features such as high question response rate, high answer quality, a spam-free environment for users and bridging disjoint social clusters. Previous works use reputation systems to achieve the goals. However, these reputation systems evaluate a user with an overall rating for all questions the user has answered regardless of the question categories, thus the reputation score does not accurately reflect the user's ability to answer a question in a specific category. We propose SmartQ: a reputation based Q&A System. SmartQ employs a category and theme based reputation management system to evaluate users' willingness and capability to answer various kinds of questions. The reputation system facilitates the forwarding of a question to favorable experts, which improves the question response rate and answer quality. SmartQ bridges disjoint social clusters by calculating reputation scores for each cluster on each question theme; SmartQ incorporates a lightweight spammer detection method to identify potential spammers. In order to reduce the loads of experts, we propose a strategy to recommend suggested answers from similar questions to each new question. Our trace-driven simulation on PeerSim demonstrates the effectiveness of SmartQ in providing good user experience. We then develop a real application of SmartQ and deploy it for use in a student group in Clemson University. The user feedback shows that SmartQ can provide high-quality answers for users in a community.

Index Terms—Question and answer system; spammer detection; reputation system; social cluster; question category

1 Introduction

Question and Answer (Q&A) systems aim to provide collaborative answering of questions by spreading messages to a group of people with registered interest in the question topic. These systems are becoming popular as they aggregate the collected contributions and assessments of all users. In Q&A systems, askers pose questions and other users answer them. Users' participation is typically motivated by various mechanisms (*e.g.,* earning points or monetary rewards). For example, in Yahoo! Answers (YA), an answerer will receive 2 points for answering a question and 10 points if his answer is selected as the best answer [1]. Social networking is also a motivation for answering in Q&A systems. The study in [2], [3] shows that a knowledge-oriented online social network (OSN) with unidirectional links is formed in YA. If user A wants to frequently visit/track all questions and answers of user B, A adds B to its contact list by building a link to B. Then, A becomes B's fan. So every user has a contact list and fan list.

Q&A systems have significantly changed the way we seek information. When compared with traditional web search engines, Q&A systems tend to provide answers to a broader range of questions attributed to everyday life situations [4]. For example, users may use Q&A systems to ask for quick hotel suggestions, or advice on his college selection from users with relevant knowledge. There are four important issues affecting the performance of a Q&A system:

- **Response rate** The questions launched by askers need to be forwarded to the potential answers who are willing to provide help. Otherwise, the askers will suffer a long delay before receiving satisfying answers.
- **Answer quality** The objective of Q&A system to return high-quality answers to the questions, thus, identifying potential experts is crucial before forwarding the questions.
- **Disjoint social cluster** If a question cannot find an answerer within a social cluster, it should be forwarded to a cluster which can solve it.
- **Spammer detection** The Q&A system should be able to identify potential spammers and prevent them from spreading trash information.

The first issue with Q&A systems is answering rate. Answerers in the OSN are willing to and able to provide more tailored and personal answers to the questioners since they are familiar with the questioners [5]. However, there are users who do not bother to give any response to the questions they receive (lazy users). Thus, it is a common case that users will not receive answers for their questions, or suffer from long delay before they receive answers. This is normally due to lack of incentives for answering questions. Analysis on Mahalo [6], a fee-based Q&A site, shows that askers are ready to pay when requesting facts that they are interested in. However, monetary reward is not practical in most free Q&A systems, and a feasible way is to filter out lazy users when choosing answerers.

The second key issue in Q&A system is answer quality. Users want to get satisfying answers to their questions, however, it is difficult to match a question to a user who has the expertise to answer it. Nam *et al.* [7] showed that altruism, business motives, learning, hobbies, and reputation score

- *Yuhua Lin and Haiying Shen are with the Department of Electrical and Computer Engineering, Clemson University, Clemson, South Carolina 29634.*
  E-mail: {yuhual, shenh}@clemson.edu

Manuscript received April 19, 2005; revised August 26, 2015.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2735442, IEEE Transactions on Big Data

are important incentives in Q&A systems. And monetary rewards are effective incentives to improve answer quality [8]. However, monetary rewards can only promote the quality of answers when the answerers have expertise in the related field. Forwarding questions to the right experts is the key solution to increase the answer quality. Existing systems evaluate the expertise of every user by a general reputation score, which is not accurate in reflecting a user's capability in answering a specify category of questions. In our design, we provide reputation scores for each user according to different question categories and themes, which are more accurate in forwarding questions to users with the right expertise.

A third issue with Q&A system is the disjoint social cluster problem. As in the knowledge-oriented OSNs, users tend to make friends with users from the same profession, or with similar interests. For example workers from the same company, friends from the same area, or with similar interests are formed into user groups (we call these user groups clusters). Social network clusters are typically centered on similar interests, careers and knowledge, thus they are usually disjoint. So a question cannot find answerers from a social cluster if users in this cluster do not have knowledge in a specific field.

The fourth issue with Q&A systems is the detection of spammers. As large Q&A systems are exposed freely to huge amounts of users, they provide an ideal environment for spammers to distribute their commercial advertisements, or malicious users to spread trash information. Existing spammer detection methods mainly focus on characterizing spam traffic [9] and network-level spammers' behavior [10], [11]. However, monitoring and analyzing the spamming features on network traffic and user behavior is expensive.

In order to answer the four key issues, we have incorporated three components in our system design: category and theme based reputation management, reputation oriented cluster bridging, lightweight spammer detection and recommending answers from existing questions. We summarize the contributions of our paper below:

- We employ a reputation management system to facilitate the forwarding of a question to favorable experts. For each question category and question theme, a user is assigned a reputation score, this reputation scores is calculated in such a way that it reflects the user's trustworthiness and willingness to answer questions on a specific category or theme. So forwarding a question to experts with high reputation will increase the probability that the question is replied to with prompt and high-quality answers.
- We calculate reputation scores for each social cluster based on its question answering records in every theme, we then solve the disjoint social cluster problem by forwarding questions to a cluster with a high reputation on the specific question theme.
- Based on the rationale from [2], [3] that a linear relationship exists between the number of best answers and the number of all answers for contributing users, we then propose a lightweight spammer detection method to identify potential spammers. This method examines the ratio of best answer count and total number of answers provided by each user (RBA), and users with low RBA will be regarded as spammers. We further improve the precision of spammer detection by studying the number of contacts a users attracts.
- To reduce the loads of experts, we propose a strategy to recommend suggested answers from similar questions to each new question. If the asker of the new question is satisfied with the suggested question, this question will not be forwarded to the experts.

The remainder of this paper is arranged as follows. Section 2 presents an overview of the related work. Section 3 presents a detailed description of SmartQ. Section 4 present the experimental results on both PeerSim and real application. Section 5 concludes this paper with remarks on future work.

2 Related work

Recent years have witnessed a rapid rise in prevalence of online Q&A systems [1], [12], [13] in our daily lives. Facebook launched a Q&A application in July, 2010 which facilitates users posting and answering questions through the OSN in order to take advantage of the collective intelligence of their friends. Early works in Q&A system research community focus on analyzing some of the large-scale Q&A sites, such as Yahoo! Answers and Google Answers. Adamic *et al.* [14] showed that in Yahoo! Answers, users share knowledge across different topic categories (*i.e.*, experts in different domains help one another). Interaction among users is highly skewed depending on the question topics, and best answers can be predicted based on reply thread length. Harper *et al.* [15] showed that questions in Q&A systems can be broadly classified into two categories, which are informational questions (*e.g.*, fact and advice) and conventional questions (*e.g.*, opinions and self-expression). In their analysis of Naver KiN, Nam *et al.* [7] studied the user behavior of answerers and found that their level of participation in contributing knowledge is highly skewed, and answerers' participation tends to be intermittent.

Besides study of the basic characteristics of Q&A systems, researchers also study different ways of improving the question answer rate and quality. Various approaches can be grouped into 3 main categories: 1) Using a centralized server to forward questions automatically; 2) Leveraging social networks for knowledge sharing; and 3) Adopting a reputation system to identify reliable answerers. Centralized Q&A systems, such as Aardvark [16] and IM-an-Expert [17], rely on a centralized server to forward questions to appropriate users in the community. However, the centralized server may suffer from high service request rate and traffic congestion. Social networks are an effective tool for facilitating knowledge sharing [18], [19]. ReferralWeb [20] and Expertise Recommender [21] both exploit the social network within a community to identify a set of experts with regards to the information in need. Also, Shah *et al.* [22] ascribed the success of the Yahoo! Answers to its reward policy, such as the levels and ranks achieved through contributing useful answers to the community, they also concluded that one reason for the failure of Google Answers [12] was its lack of a social component. Lee *et al.* [23] pointed out that

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2735442, IEEE Transactions on Big Data

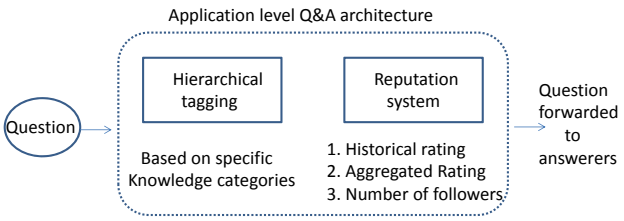


Fig. 1: An overview of distributed Q&A system.

answerers are mainly motivated by financial incentives and intrinsic motives. Thus, besides incentives and rewards, forwarding questions to users with matched expertise is crucial to improve Q&A performance. Some works apply a reputation system to locate credible answerers [24], [25], these systems maintain a general reputation score for every user as an indicator of whether the user reliably provides high-quality answers. SmartQ is distinguished from these works in a way that it provide reputation scores for each user according to different question categories and themes, which help to navigate questions to the right experts and improves question response rate and answer quality.

Also, a Q&A system needs to be clean and user-friendly to earn user loyalty, and various methods have been proposed to assess whether a user is contributing relevant and useful information. Pelechrinis *et al.* [26] proposed a collaborative assessment method to identify spammer in Q&A system, where each user monitors the activity of other users and observes their compliance with predefined cognitive models. Long *et al.* [27] proposed a collaborative filtering method to limit spammer hazards, which calculates the importance score of each user based on his/her relationships to other users. However, these schemes are not robust to the presence of malicious entities, as they do not consider the correctness of the subjective feedback from users. SmartQ incorporates a lightweight spammer detection method to keep the Q&A system clean and prevent the dissemination of useless information.

3 System Design

Q&A systems have created abundant resources of millions of questions and hundreds of millions of answers. This paper proposes a novel reputation system for computing user reputation score as a reflection of the answerer's willingness to reply and trustworthiness of the response information.

3.1 An overview of SmartQ

When a user launches a new question, this question is labeled with tags describing the question's category and theme. Then the question is forwarded to its contacts who are registered with interests in the according category and theme. In order to improve the chance of getting satisfying answers, the question should be forwarded to users who are experts in the field of the question, and are willing to answer the question. We assign every user with reputation scores as indications of the user's ability and willingness to answer questions. The reputation scores are estimated by considering three factors. 1)Direct trust, which is calculated by examining the historical interaction records between two users. A user's reputation score is accumulated by serving incoming questions with valuable answers throughout a long time period. We assign different weights to answering activities based on their timestamps. 2)Aggregated trust, which is calculated by gathering the opinions from a user's fans. In large-scale online systems such as Q&A systems, a user only interacts with a subset of all users (*i.e.*, a user's fans) [28], and a user tends to trust its fans' opinions. So a user only aggregates opinions from its fans when calculating another user's reputation. 3)Trustworthiness, which is evaluated by the number of fans a user attracts. The first two factors consider the qualities of answers a user provides, by studying the interaction experience between users. While the third one relies on the fact that user A connects to B only when user A trusts B's knowledge, as A believes that B is capable of answering its questions. The reputation system helps to identify a list of users who are likely to provide high-quality answers for each question, then the system forwards the question to a number of potential answerers.

An overview of SmartQ is shown in Figure 1. When user A launches a question at time t, it defines the question's main category and detailed theme in the hierarchical tagging stage. Then we compare the reputation rating of A' contacts, and select a number of contacts with high reputation scores. The question is then forwarded to the highly regarded contacts. The reputation system is responsible for updating users' reputation ratings at a specific frequency. Important notations used in this paper are listed in Table 1.

TABLE 1: Table of major notations.

q_k	a question
c_u	a category
t_{uv}	a theme belonging to category c_u
R^c_{ab}	ratings of user B given by A in different categories
r^c_{abu}	user B's reputation given by A on category u
R^t_{ab}	ratings of user B given by A in different themes
r^t_{abv}	user B's reputation given by A on theme v
s_{abk}	rating on question q_k user A gives user B
$\mathcal{Q}^u_b$	set of questions on category u answered by user B
$\mathcal{T}_u$	a set including all question themes under category c_u
$\lvert\mathcal{T}_u\rvert$	the number of themes in category c_u
F_a	a set containing all user A's fans
Z_{abk}	overall reputation of B given by A on question q_k
C_a	a social cluster
r^t_a	the reputation of cluster A on various of themes
$\mathcal{Q}^v_a$	questions answered by users in cluster A on theme v
s_k	score earned by answering question q_k
N_a	number of all answers provided by a user
N^b	number of best answers for a user
R	best answer rate for a user
f_{t,q_i}	raw frequency of term t in question q_i
$tf(t,q_i)$	term frequency of t in question q_i
$idf_(t,q_i)$	inverse document frequency of t in question q_i
$tfidf(t,q_i)$	term frequency-inverse document frequency of t
N_q	total number of questions in the system
ST_{ij}	similarity between question titles of q_i and q_j
SD_{ij}	similarity between question descriptions of q_i and q_j
CS_{ij}	overall cosine similarity between question q_i and q_j
a_{ij}	an answer j to question q_i
AL_{ij}	length of answer a_{ij}
$\mathcal{AL}_{ij}$	normalized value of AL_{ij}
$\mathcal{RBA}_{ij}$	best answer rate of the answerer of a_{ij}
QS_{ij}	quality score of answer a_{ij}

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2735442, IEEE Transactions on Big Data

3.2 Question Category Selection

Popular Q&A systems can generate large amount of questions every day, grouping and organizing the questions by their specific knowledge area is crucial to help users find the questions they are interested in. In SmartQ, we assign two levels of tags to questions: category and theme. Category is a larger domain than theme, and every category contains at least one theme. For example, sports, literature and movie are categories a question may belong to; under sports category, there are multiple themes such as soccer, football and basketball. We use q_k to denote a question; c_u be a category; t_{uv} be a theme belonging to category c_u. Thus, a question belonging to category c_u is denoted by $q \in c_u$, and a question belonging to theme t_{uv} is denoted by $q \in t_{uv}$.

3.3 Category and Theme based Reputation Management of Users

Given $q_k \in c_u$ and $q_k \in t_{uv}$, user A evaluates user B's reputation on question q_k based on direct trust, aggregated trust and trustworthiness. Direct trust is evaluated based on A's experience; aggregated trust is calculated by gathering opinions from all A's fans; and trustworthiness is measured by examining the number of B's fans.

3.3.1 Direct trust

In direct trust, user B's reputation is calculated based on past interactions (*i.e.*, the answers A receives from B). Direct trust is expressed by two factors, category reputation and theme reputation, which are represented by two vectors. $R^c_{ab} = (r^c_{ab1}, r^c_{ab2}, ...r^c_{abn})$ stores the rating of B in different categories, while $R^t_{ab} = (r^t_{ab1}, r^t_{ab2}, ...r^t_{abm})$ represents the rating of B in different themes. Every element in R^c_{ab} and R^t_{ab} is calculated by summarizing the questions belonging to a specific category or theme. In order to reflect a user's recent performance, his recent answering behaviors are assigned with higher weight in reputation calculation. We apply an exponential decay factor $\phi_k \in [0, 1]$ for question q_k, ϕ_k is initialized to 1 and decreases as time elapses. $\phi_k = e^{-\lambda t_k}$, and t_k is the time period that question q_k has been answered. Then, user B's reputation on category u is calculated by:

$$r^c_{abu} = \sum_{q_k \in \mathcal{Q}^u_b} \phi_k \times s_{abk}, \tag{1}$$

where s_{abk} is the rating on question q_k user A gives user B, and $\mathcal{Q}^u_b$ is a set of questions on category u that are answered by user B. The theme reputation r^t_{ab} is calculated in the same way as that in Equation (1), i.e., $r^t_{abv} = \sum_{q_k \in \mathcal{Q}^v_b} \phi_k \times s_{abk}$. After A calculates B's category reputation vector R^c_{ab}, and theme reputation vector R^t_{ab}, the two vectors are sent to all A's contacts. Suppose user L is one of A's contacts, R^c_{ab} and R^t_{ab} are needed by user L to calculate user B's aggregated trust value. Note that the information of direct trust does not need to be exchanged on a regular basis, when there are updates of A's direct trust towards another user, A needs to send this update information to all his/her fans. In order to make the exchange of reputation values safe and accurate, SmartQ should withstand some common types of network attacks, such as Man-in-the-Middle attack and data modification. Various approaches such as PKI [29] have been well-developed to prevent these attacks, so this issue is not the focus of our paper. Finally, the direct trust of A towards B regarding question q_k is calculated by:

$$r_{abk} = 1/|\mathcal{T}_u| \sum_{p \in (\mathcal{T}_u \setminus t_{uv})} \Lambda_p \times r^t_{abp} + \gamma \times r^c_{abu}. \tag{2}$$

In Equation (2), $\mathcal{T}_u$ is a set including all question themes under category c_u, and $|\mathcal{T}_u|$ is the number of themes in category c_u. $\gamma \in (0, 1)$ is the weight of category reputation, $\Lambda_p \in (0, 1)$ is the weight of theme reputation for theme t_{up}.

3.3.2 Aggregated trust

In aggregated trust, a user listens to his/her fans' opinions when evaluating another user's reputation. User A receives category reputation vectors and theme reputation vectors of user B from all its fans. The aggregated ratings of B in different categories and themes are stored in vector $r^{c'}_{ab}$ and $r^{t'}_{ab}$, respectively. In the following, for simplicity, we use r^x_{ab} to represent both r^c_{ab} and r^t_{ab}, and use $r^{x'}_{ab}$ to represent both $r^{c'}_{ab}$ and $r^{t'}_{ab}$. When A wants to compute the aggregated reputation of user B on category or theme y, A sums all B's category or theme reputations received from its fans weighted by the closeness between them in Equation (3).

$$r^{x'}_{aby} = \frac{\sum_{d \in F_a} \Theta_{ad} \times r^x_{dby}}{\sum_{d \in F_a} \Theta_{ad}}. \tag{3}$$

Where r^x_{dby} is the direct rating of user D towards user B on category or theme y. F_a is the set containing all user A's fans. Θ_{ad} is the weight of closeness between user A and D, $\Theta_{ad} \in (0, 1)$. Similar to Equation (2), the aggregated trust of user A towards B is r'_{ab} calculated by:

$$r'_{abk} = 1/|\mathcal{T}_u| \times \sum_{p \in (\mathcal{T}_u \setminus t_{uv})} \Lambda_p \times r^{t'}_{abp} + \gamma \times r^{c'}_{abu}. \tag{4}$$

3.3.3 Overall reputation

Finally, user A calculates the overall reputation of B with respect to question q_k (denoted by Z_{abk}) in Equation (5).

$$Z_{abk} = \alpha \times r_{abk} + (1 - \alpha) \times r'_{abk} + \beta(f_b/\Phi) \tag{5}$$

Where $\alpha \in [0, 1]$ is the weight placed on direct trust, large value of α means that user want to evaluate another users reputation mainly based on its own experience. f_b is the number of fans B attracts, and Φ is the total number of users in the system. The first two elements in Equation (5) consider the quality of answers provided by B, while the third element consider the general reputation of B. $\beta \in (0, 1)$ is the weight assigned on general reputation.

When the system needs to forward question q_k asked by user A to its contacts, the system examines the reputation of all its contacts with respect to question q_k. An example of calculating user A's reputation is shown in Figure 2. Suppose q_k is in category c_u and theme t_{uv}. User A first calculates its direct trust to B based on historical records in step 1, which includes category trust r^c_{abu} and theme trust r^t_{abv}. In step 2, A gathers direct trust of all his fans towards user B, then calculate B's aggregated trust $r^{c'}_{abu}$, and $r^{t'}_{abv}$. Finally, user A calculate an overall reputation for B on question q_k in step 3.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2735442, IEEE Transactions on Big Data

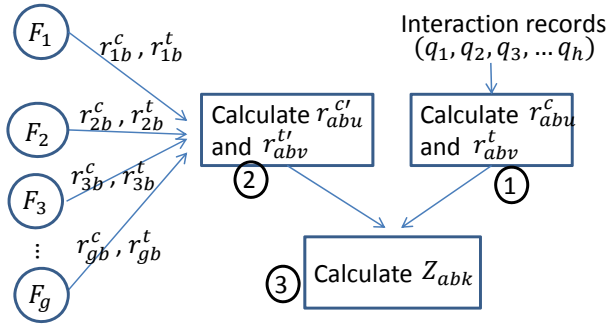


Fig. 2: An example of reputation calculation.

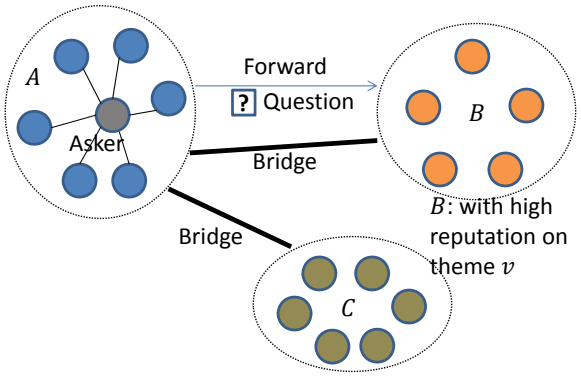


Fig. 3: An overview of bridging disjoint clusters.

3.4 Encouraging User Participation with Reputations

An important goal for online communities and social media sites is to attract more users and encourage user engagement. To achieve this goal, a reward system called badges have been widely used in various websites, including educational sites like Khan Academy and knowledge-creation Q&A sites like Stack Overflow. Badges are public reward to recognize users' contributions to the community. A greater number of badges earned by a user in some way indicate that this user is a highly reliable and trustable expert, which intuitively motivates users to actively answer new questions and earn as many badges as possible.

In SmartQ, reputation scores are used to award users who provide high-quality answers for questions, which are perceived as users' expertise and respect in the community. A user needs to contribute a substantial amount of work in answering questions in order to earn a high reputation score. Reputation scores are similar tools as badges, they act as useful incentives in motivating users to participate in the question answering activities. Reputation scores can work alongside with badges to serve as a summary of a user's key accomplishment.

3.5 Reputation Oriented Cluster Bridging

Social clusters on Q&A systems are generally formed by friends from the same area, or with similar interests. Within each cluster, users tend to have similar background, knowledge and life experience. The study in [2], [3] indicates that, in the knowledge-oriented OSNs, some of the social network clusters specialized on different categories are likely to be disjointed. For example, staffs from the same company may form a social cluster, whose knowledge comprises highly specialized insights, information and experiences about the company's field of business. However, people from this social cluster may not be familiar with other fields of business. Therefore, a user may not receive answers for certain questions in the Q&A system because his interacted users have small knowledge base, and the user's question cannot reach other parts of the Q&A system. Therefore, we need to create bridges between different social clusters to prevent the isolation of some users' social network.

In this design, we provide a strategy to navigate a question to find the desired answerers when it cannot be solved within a social cluster. For each social cluster C_a, we associate it with a theme reputation vector, $r^t_a = (r^t_{a1}, r^t_{a2}, \dots r^t_{an})$. r^t_a records the reputation of cluster A on various of themes. Each element in r^t_a is calculated by summing the answer scores earned by all the cluster members, which is shown in Equation (6).

$$r^t_{av} = \sum_{q_k \in \mathcal{Q}^v_a} \phi_k \times s_k. \tag{6}$$

$\phi_k \in [0, 1]$ is an exponential decay factor defined in Equation 1, $\mathcal{Q}^v_a$ is a set of questions that are answered by users from cluster A on theme v, and s_k is the score earned by answering question q_k. Then the reputation of each cluster on answering questions from a specific theme is calculated. Note that we calculate a cluster's reputation based on each question theme, as theme is a finer granularity than category in grouping questions, so a reputation score calculated based on each question theme can reflect a cluster's ability in answering a specific type of questions. Shown in Figure 3, when forwarding a question q_k in theme v, we select a cluster with the highest reputation on theme v. Inside each cluster, when question q_k arrives, the system forwards it to a user with the highest reputation on the question's theme.

To further improve the speed of question forwarding, for a social cluster A, we define a set of close clusters $\mathcal{M}_a$. If a question from one cluster cannot be solved within the cluster, it will be forwarded to close clusters. We use cluster closeness ($\mathcal{CC}$) of two clusters to reflect the frequency of interactions (forwarding and answering questions) between these clusters. For a question q_k launched by users in cluster A, if an answer is received from cluster B, we call it a hit $I^k_{ab} = 1$, thus the closeness of cluster A to B is calculated by summarizing all hits: $\mathcal{CC}_{ab} = \sum_{\forall q_k} I^k_{ab}$. If $\mathcal{CC}_{ab} > \eta$, it means that users in B frequently help A with unsolved questions. Thus, A builds a bridge to B, and B is added into set $\mathcal{M}_a$. Next time when there is an unsolved question from A, it will be forwarded to clusters in $\mathcal{M}_a$. In the example of Figure 3, cluster A builds bridges to both B and C.

3.6 Lightweight Spammer Detection

In online Q&A systems, every registered user can post questions and answers. Spammers can take advantage of this free environment and popularity of Q&A systems, and post commercial spam to gain attention for their products. Spammers are detrimental to the Q&A systems as they do not contribute useful information. They post advertisements or other irrelevant answers aiming at spreading advertise

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2735442, IEEE Transactions on Big Data

or achieving other goals. Some spammers directly publish content to answer questions asked by common users. Additionally, another kind of spammers (we refer them as"best answer spammers") create multiple user accounts, and use some accounts to ask a question, the others to provide answers which are selected as the best answers by themselves. They deliberately organize themselves in order to deceive readers. These kind of spammers are even more hazardous, since they are neither easily ignored nor identifiable by a human reader. Google Confucius Community Question Answering system also reported that best answer spammers may generate amounts of fake best answers [30], which could have a non-trivial impact on the quality of machine learning model. Thus, identifying spammers quickly and precisely is crucial to maintain healthy development in Q&A systems.

Non-textual features such as the ratio of best answers, the number of points and the number of friends are crucial criteria to estimate the contribution of users, which can be utilized to evaluate where a user is a spammer or not. Users who are regarded as contributed members in the Q&A systems are not likely to be spammers. In this lightweight spammer detection strategy, we take two non-textual features (i.e., the ratio of best answers (R) and the number of friends) as an example and use them to determine if a user is a spammer. To additionally consider other features, we can directly add them in the similar manner.

Study in [2], [3] shows that a linear relationship exists between the number of best answers and the number of all answers of a user with correlation coefficient equals 0.712. A spammer tends to post many answers (which are in fact spam), and few of which would be selected as best answers. To determine if user A is a spammer or not, we can examine the ratio (R) of number of best answers (N^b) and the number of all answers (N_a): $R_a = N_a^b/N_a$. Given a predetermined threshold ξ, if $R_a < \xi$, user A will be identified as suspected spammer. Although spammers can collude to rate their own answers as best answers, thus increasing the ratio R. However, as the best answers are highlighted in the Q&A forum with high visibility to many other users, the false best answers can be easily identified using the abuse report policy.

Algorithm 1 Pseudo-code of Spammer detection algorithm.

```
Input: N_a^b, N_a, f_a;
Output: a list of suspected spammers SU;
for each user A do //iterate all users
    calculate best answer rate R_a = N_a^b/N_a
    if R_a < ξ then //best answer rate below threshold ξ
        if f_a < τ then //number of fans below threshold τ
            add user A to suspected spammers list SU
        end if
    end if
end for
return suspected spammers list SU
```

The value of ξ should be determined carefully to provide good performance of this spammer detection method. ξ needs to be set large enough to maintain a good detection precision. However, some users who are not knowledgeable enough to contribute a minimum ratio of best answers will be falsely identified as spammers, thus increasing the detection false positive rate. To solve this problem, we further propose an incremental strategy to reduce the chance of falsely identifying a normal user as a spammer, which considers the user's social relationship. From [2], [3], we see that a user with higher rank is likely to have larger number of fans, and the number of fans of all users follows a power-law distribution. We first define a fan count threshold τ, which is a certain percentile of the number of fans of all users in the system. If a user is in the contact lists of a large number of users, the user is not likely to be a spammer. Then we compare τ with the number of fans (f_a) that user A has, if $f_a < \tau$, user A is likely to be a spammer. The detailed spammer detection algorithm is shown in Algorithm 1.

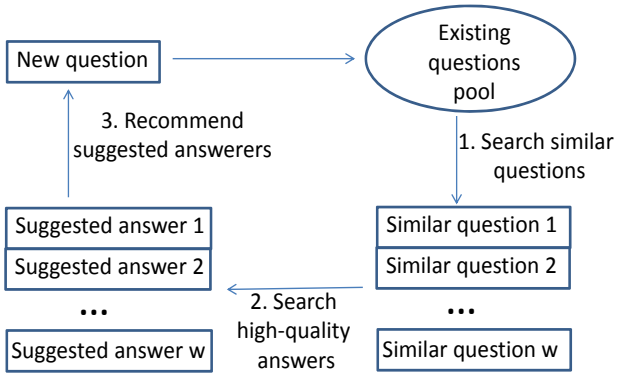


Fig. 4: Process of recommending suggested answers for a new question.

3.7 Recommending Suggested Answers to Reduce Loads on Experts

A large portion of users in the Q&A system cannot provide satisfying answers to newly generated questions, either because they are reluctant to help or they do not have the expertise to answer those questions; while a small number of users actively contribute to providing satisfying answers. Therefore, forwarding new questions to selfless experts enables askers to be more likely to receive satisfying answers. However, these selfless experts may receive an excessive number of questions frequently and do not have enough capacity and time to handle all incoming questions. In order to reduce the loads of experts, we can suggest answers from similar questions to each new question considering the fact that some users ask similar questions.

In Q&A systems, there are a limited number of hot themes that users are most interested in. Within these themes, users may repeatedly ask similar questions. For example, smartphone users may ask for instructions about how to setup email accounts on their phones. A well-developed Q&A system should be able to quickly answer these questions, which can improve the answering rate and reduce the loads of experts. In our strategy, when a user asks a new question, the system recommends a number of suggested answers to this question. Figure 4 shows the process of recommending suggested answers for a new question. In the first step, we search existing questions stored in the Q&A system and identify a number of similar questions. In the second step, we select suggested answers with high answer

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2735442, IEEE Transactions on Big Data

quality from these similar questions. In the third step, we recommend the suggested answers to the asker of the new question. If the asker is satisfied with the suggested answers, he/she will accept the suggested answers and notify the Q&A system. The question will not be forwarded to other users. As a result, the askers can receive answers within a short latency and the loads of experts are reduced. The first step of our strategy is introduced in Section 3.7.1 while the second step is introduced in Section 3.7.2.

3.7.1 Identifying Similar Questions

To suggest answers to a newly generated user question, we need to identify similar questions that are recorded in the system. Giving a new question q_i, we aim to identify top k similar questions of q_i from existing records. In this strategy, we utilize the textual features of the question to find matched questions as similar questions.

In Q&A systems like Yahoo! Answers and StackOverFlow, a question is composed of two parts: a brief title of the question and a relatively long description depicting the details of the question. While studies in [1], [31] show that the question title is the most effective tool to search similar questions, we also leverage the question description in order to improve the match results. To calculate the relevance between question q_i and q_j, we adopt the cosine similarity measurement, which is to calculate the ratio of common keywords to all keywords used in both questions. As term frequency-inverse document frequency (TF-IDF) is an crucial index to evaluate the importance of keywords, we weight the keywords with TF-IDF when calculating the cosine similarity between two questions. We adopt the TF-IDF calculation method that is widely used in existing works [32], which is introduced below.

We denote the raw frequency of term t in question q_i's title or description by f_{t,q_i}, the logarithmically scaled term frequency (denoted by $tf(t,q_i)$) is calculated by:

$$tf(t,q_i) = 1 + logf_{t,q_i}. \tag{7}$$

We denote the inverse document frequency by $idf_(t,q_i)$, which is to evaluate how much information a term can provides. Assume the total number of questions is N_q, the inverse ratio of questions containing the term t is $N/|f_{t,q_i}$. $idf_(t,q_i)$ is calculated as the logarithmically scaled of $N/|f_{t,q_i}$ in Equation (8). $idf_(t,q_i)$ evaluates whether a term is widely used or rarely used across all questions. For some common terms like stop-words (i.e., words like "is" and "are" that contain no semantic information), their $idf_(t,q_i)$ values are low as they are widely used in all questions.

$$idf_(t,q_i) = logN/|f_{t,q_i}|. \tag{8}$$

After we get both term frequency and inverse document frequency of term t, the TF-IDF value of t in question q_i is calculated as:

$$tfidf(t,q_i) = tf(t,q_i) \times idf(t,q_i). \tag{9}$$

The cosine similarity measurement is applied to calculate both the cosine similarity between question q_i and q_j. In our design, we leverage both question titles and question descriptions when calculating the similarity between two questions. We use ST_{ij} to denote the similarity between question titles of q_i and q_j; and use SD_{ij} to denote the similarity between question descriptions of q_i and q_j. Assume a list of common keywords used in the titles of both q_i and q_j are stored in $TT_{ij} = (t_1, t_2, ..t_m)$, ST_{ij} is calculated by:

$$ST_{ij} = \frac{\sum_{k=1}^{m} tfidf(t_k,q_i) \times tfidf(t_k,q_j)}{\sqrt{\sum_{k=1}^{m} tfidf(t_k,q_i)^2}\sqrt{\sum_{k=1}^{m} tfidf(t_k,q_j)^2}}. \tag{10}$$

Each term t_k is a keyword in TT_{ij}. Note that we exclude the stop-words and only consider ordinary works in TT_{ij}. Similarly, we can calculate the cosine similarity (SD_{ij}) between question descriptions of q_i and q_j. Cosine Similarity ST_{ij} is a metric to evaluate how relevant the two questions are, which takes into consideration of the TF-IDF weight of each keyword of each question.

The overall cosine similarity between question q_i and q_j (denoted by CS_{ij}) is calculated by:

$$CS_{ij} = \alpha ST_{ij} + (1-\alpha)SD_{ij}, \tag{11}$$

where α is the weight placed on the the similarity between question titles of q_i and q_j. As question title is more effective in searching similar questions, we assign heavier weight on ST_{ij}, i.e., $\alpha > 0.5$. After we calculate the similarities of all existing questions to a newly generated question q_i, we rank the questions based on their similarity scores. We then select the top w questions with highest CS_{ij} values as similar questions, which are stored in a list $QS(q_i) = (q_1, 1_2, ...q_w)$. In the next section, we will introduce how to identify suggested answers from these similar questions.

Algorithm 2 Pseudocode of selecting top w similar questions.

```
1: Input: new question q_i; list of existing questions Q;
2: Output: top w similar questions QS(q_i) = (q_1, 1_2, ...q_w);
3: for each q_j ∈ Q do //iterate all questions in Q
4:     for each keyword t_i in q_j's title do //iterate keywords in the title
5:         calculate the term frequency tf(t_i, q_i) = 1 + logf_{t_i,q_i}
6:         calculate idf value idf(t_i, q_i) = logN/|f_{t_i,q_i}|
7:         calculate the TF-IDF value tfidf(t_i, q_i) = tf(t_i, q_i) × idf(t_i, q_i)
8:     end for
9:     calculate ST_ij between titles of q_i and q_j using Equation (10)
10:    for each keyword t_j in q_j's description do //iterate keywords in the description
11:        calculate the term frequency tf(t_j, q_i) = 1 + logf_{t_j,q_i}
12:        calculate idf value idf(t_j, q_i) = logN/|f_{t_j,q_i}|
13:        calculate the TF-IDF value tfidf(t_j, q_i) = tf(t_j, q_i) × idf(t_j, q_i)
14:    end for
15:    calculate SD_ij between descriptions of q_i and q_j
16:    calculate overall similarity CS_ij using Equation (11)
17: end for
18: sort all questions in Q based on CS_ij values
19: select top w questions in Q and store them in QS(q_i)
20: return QS(q_i)
```

Algorithm 2 shows the pseudocode of selecting top w similar questions for a new question q_i. The algorithm first iterates all keywords in the title of an existing question q_j and calculates each keyword's term frequency, inverse document frequency and TF-IDF values (lines 4-8). It then calculates the similarity score between the titles of q_i and q_j using Equation (10) (line 9). Next, the algorithm calculates the term frequency, inverse document frequency and TF-IDF values of each keyword in the description of q_j (lines 10-14). It also calculates the similarity score between the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2735442, IEEE Transactions on Big Data

descriptions of q_i and q_j (line 15) and the overall similarity CS_{ij} using Equation (11). Finally, the algorithm sorts all questions in Q based on CS_{ij} values and returns the top w similar questions. Sorting all question in Q requires a computational complexity of $O(N_q \log N_q)$, where N_q is the number of questions. Assuming the total number of keywords in a question's title and description is far less than n, the computational complexity of Algorithm 2 is $O(N_q \log N_q)$.

3.7.2 Recommending Suggested Answers

In the previous section, we introduced how to find the questions in the records that are semantically similar to the new question. Next, we need to find high-quality answers from the answers of these identified similar questions (called candidate questions) as suggested answers for a new question. In this section, we introduce how to identify the high-quality answers. In Q&A systems, a question usually attracts multiple answers of different quality from other users. For each candidate question, we need to select an answer with the highest quality, we then forward the answer together with the question to the asker. The selected answer of a candidate question comes from two sources: the best answer selected by the asker; if no best answers are selected by the asker, we need to select a preferred answer that is evaluated to have high quality.

Given a candidate question, we first try to identify the best answer to each candidate question. As the best answers chosen by the Q&A system community may not truly reflect the quality of the answers [33], we only choose the best answer nominated by the askers. If there is no best answer selected by the asker of the candidate question, we select a preferred answer that is evaluated to have high quality (i.e., an answer that is likely to solve the question). To select a preferred answer for a candidate question, we develop an answer quality evaluation method relying on the features of user-generated answers.

Prior works reveal that some textual and non-textual features are effective tools to evaluate the quality of answers [34], [35], which include answer length, answerer's ratio of best answers, answerer's activity level, answerer's category specialty, and so on. In our design, we utilize two most important features to estimate the quality of questions:

- Answer length (AL). AL is the number of words in an answer, which is the most significant factor in predicting the quality of answer [36]. In AL, we filter out the commonly occurring stop-words and only consider the unique non-stop words.
- Answerer's ratio of best answers (RBA). RBA is the ratio of best answers to all answers provided by the answerers, which is a crucial factor to evaluate the expertise of the answerer. RBA is leveraged based on the assumption that an answerer used to contribute good answers is likely to maintain his/her performance.

Suppose a_{ij} is an answer for question q_i and its length is AL_{ij} and the best answer rate of its answerer is $\mathcal{RBA}_{ij}$. We first normalize the value of AL_{ij} to 1. $\mathcal{AL}_{ij} = AL_{ij}/AL_{80}$, where AL_{80} is the 80th percentile AL value of all answers in the Q&A system. We set $\mathcal{AL}_{ij}$ to 1 if its value is greater than 1. We then calculate the quality score of a_{ij} using Equation 12.

$$QS_{ij} = \alpha \mathcal{AL}_{ij} + (1-\alpha)\mathcal{RBA}_{ij}, \tag{12}$$

in which we use the weighted average value of two quality factors. As AL is a more significant factor than BAR when determining the quality of answers, the value of α is higher than 0.5 in order to place more weight to AL. For each candidate similar question q_i, we calculate quality scores for all answers of q_i, we then select the answer with the highest score and forward q_i with its best answer to the asker.

Algorithm 3 Pseudocode of selecting suggested answers based on answer quality.

```
1: Input: similar question q_i; list of answers for q_i;
2: Output: a suggested answer a_ij;
3: if q_i has a best answer a_ij then //best answer exists
4:    return a_ij //best answer is selected as the suggested answer
5: end if
6: for each answer a_ij of q_i do //iterate all answers
7:       calculate the normalized answer length AL_ij
8:       calculate quality score of a_ij using Equation 12
9: end for
10: sort all answers of q_i based on quality scores
11: select answer a_ij with the highest quality score
12: return a_ij
```

Algorithm 3 shows the pseudocode of selecting suggested answers for a similar question q_i. The algorithm first checks if the asker of q_i has selected a best answer among all answers of q_i (line 3). If the best answer exists, the algorithm returns the best answer as the suggested answer (line 4). Otherwise, it iterates all answers and calculates the qualify scores (lines 6-9). Next, the algorithm sorts all answers of q_i based on the quality scores and returns the answer with the highest quality score. As sorting all answers of q_i needs a computational complexity of $O(n \log n)$, where n is the number of answers, the computational complexity of Algorithm 3 is $O(n \log n)$.

4 Performance Evaluation

We conducted trace-driven experiments on PeerSim [37]. The data set we used is crawled from *Yahoo! Answers* from Aug. 17 to Oct. 19, 2011, which includes: 1) personal information of 119,175 users such as best answer rate (which is the percentage of a user's answers that are chosen by the askers as best answers), number of followers (i.e., fans) and contacts for each users, 2) general information of 119,174 questions such as the categories they belong to and the answers they draw. According to *Yahoo! Answers*, the questions are grouped by 26 categories, including "Travel", "Environment", and 148 themes including "Air Travel" and "Australia" under category "Travel". In the simulation, we deployed 10,000 nodes as users on Q&A system; the users are selected from the trace data who have more than 6 contacts. Follower and contact relationships are set based on user information from the trace data. Each user has 1 to 4 randomly selected question categories (interests), and has 1 to 5 themes under each question category. The expertise level of each user in a category or a theme is chosen from 1 to 10. The expertise level indicates a user's ability to answer questions, higher level in a specific question

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2735442, IEEE Transactions on Big Data

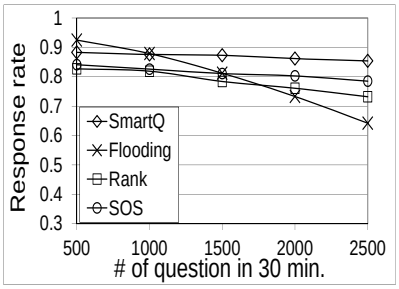


Fig. 5: The question response rate.

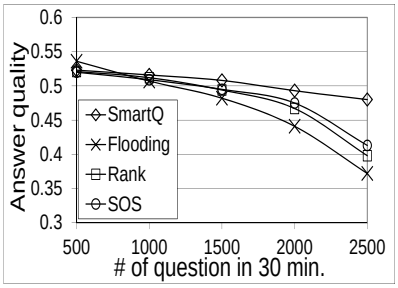


Fig. 6: The average quality of answers.

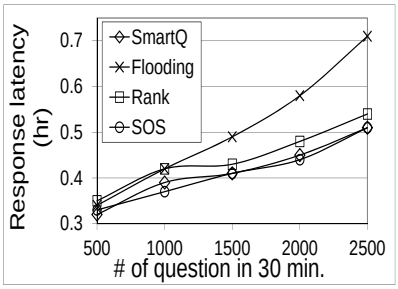


Fig. 7: Average latency of receiving answers.

theme represents higher proficiency in answering questions belonging to the theme. In order to have more capable answerers in the system, in additional to v number actual answerers in the trace, we also randomly selected $10v$ users from the users who have interest in the question's theme as capable answerers. After receiving a question, if a user is a capable answerer of this question, (s)he will respond after a delay randomly chosen from [1,30] minutes. A user can answer up to 2 questions within every 30 minutes. An asker will rate each answer with scores based on the answerer's expertise level. If an answer is received from a user with level l expertise, then the asker will rate this answer with $l/10$ score. In order to generate answering activities and cumulate reputation scores for users, we executed a warm-up process by launching 20,000 questions. During the test, user reputation was updated every 30 minutes. Λ_p and γ in Equation (2) are set to 0.4 and 0.6, respectively; α and β in Equation (5) are to 0.7 and 0.5, respectively; other parameters are set as λ=1, τ=10 and η=100. The simulation contains a 12 hours process, within every 30 minutes, a number of randomly users post questions and these questions are forwarded to their contacts.

In our proposed SmartQ Q&A system, when a user posts a question, the contacts of this user are sorted by their reputation scores on the question's theme and category. The question is forwarded to 3 contacts with the highest reputation scores. If no answer returns after 30 minutes, this question is forwarded to the next 3 contacts, and then the question forwarding operation is terminated. We compared our proposed *SmartQ* Q&A system with three strategies. In the *Flooding* strategy, a user's question is broadcasted to all its contacts; In *Rank*, a user's question is forwarded to the user with 3 highest best answer rate among its contacts; *SOS* [38] forwards a user's question to 3 contacts who have the highest similarity value to the asker, and the similarity is measured by examining the contact's interests and social closeness to the asker. Similar to *SmartQ*, if no answer returns after 30 minutes, *Rank* and *SOS* will forward a question to the next 3 contacts with high best answer rate and similarity value to the asker, respectively, and then the question forwarding operation is terminated. We are interested in the following metrics:

- **Response rate** The percentage of questions that can receive at least one answer [39].
- **Answer quality** The rating of answers given by the askers.
- **Response latency** The time spans from a question is launched until it draws the first answer.
- **Overhead** Number of question forwarding actions executed by the system.

4.1 Simulation Results

We examine the overall performance of *SmartQ* in terms of all interested metrics.

Figure 5 shows the question response rate when there are different numbers of new questions posted in the system within 30 minutes. We see that *Flooding* achieves the best response rate at around 0.85 when the new question arrival rate is small. The response rate drops gradually when the question arrival rate increases, as users do not have enough capacity to answer all new questions. *Rank* yields the least response rate as new questions are always forwarded to users with high best answer rate, and these users are not capable of providing answers to all new questions. Also, users with high best answer rate may not have the expertise to answer a specific question. *SOS* outperforms *Flooding* and *Rank* due to the reason that *SOS* locates potential answerers by examining the closeness of user interests and a question's category. *SmartQ* is effective in providing high question response rate under different question arrival rates, due to the reason that the question is forwarded to a limited number of users with expertise in the question's specific area.

Figure 6 shows the average answer quality when new questions are posted in the system at different rates, which is evaluated by averaging all answer scores received from askers. If no answer is provided by an asker, the score for this answer quality is 0. We see that *SmartQ* is advantageous in maintaining high answer quality at about 0.5, due to the reason that the question is forwarded to potential answerers with high reputation in the question's specific area. *SOS* achieves higher answer quality than *Flooding* and *Rank* as it studies the similarity between question category and user interests. However, a user's interest in a question category does not guarantee sufficient expertise in solving questions in this category, thus *SOS* gets lower answer quality than *SmartQ*. *Flooding* and *Rank* both do not consider the users' ability and willingness to answer a specific question, so they cannot provide high answer quality for askers.

Figure 7 shows the average latency of receiving answers, which is measure from the time a question is launched until the time the first answer arrives. We see that as questions are posted at a higher rate in the system, the average latency

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2735442, IEEE Transactions on Big Data

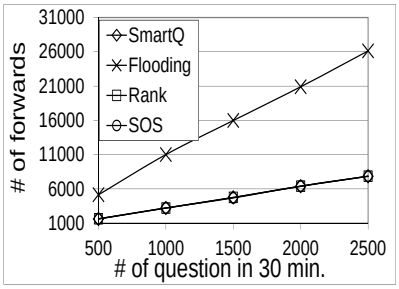


Fig. 8: # of forwards.

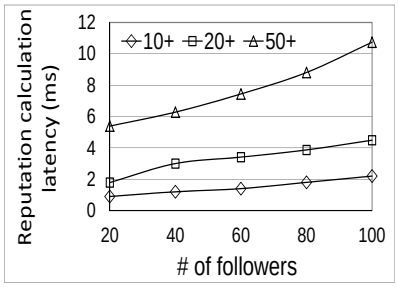


Fig. 9: Average reputation calculation latency.

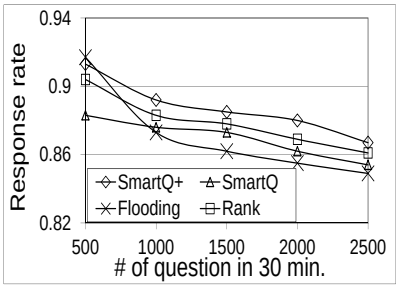


Fig. 10: The question response rate.

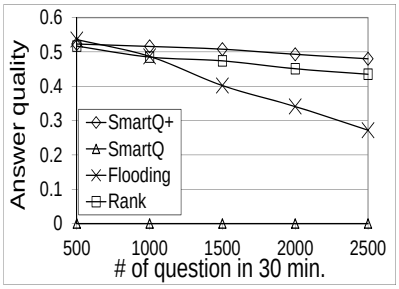


Fig. 11: The average quality of answers.

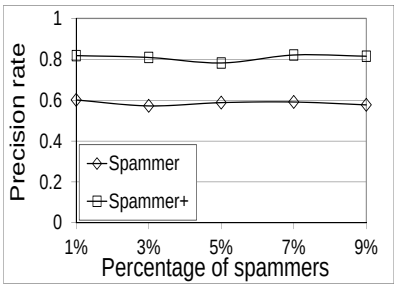


Fig. 12: Precision rate of spammer detection strategy.

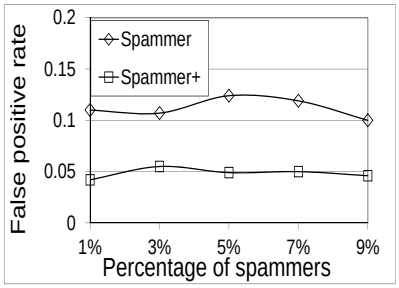


Fig. 13: False positive rate of spammer detection strategy.

of receiving answers increases for all strategies, due to the reason that users can provide a limited number of answers within every time period. We also see that both *SmartQ* and *SOS* outperform other two strategies in reducing answering latency, as they both explore users' expertise or interests while forwarding questions. *Flooding* can easily overwhelm users with excessive number of questions, thus the average answering latency is increased.

Figure 8 shows the total number of forward actions executed with different new questions arrival rates. We see that *SmartQ*, *Rank* and *SOS* need less number of forward actions than *Flooding*, as they only forward a new question to at most 6 users.

Figure 9 shows the computation cost of reputation calculation when there are different numbers of followers for each user. Three lines represent calculation time for different number of users. We see that the latency of reputation calculation is generally short, and it takes about 11ms to finish reputation calculation of 50 users. Note that the calculation time can be further reduced by parallelism. Figure 9 indicates that *SmartQ* is able to execute fast question question forwarding actions.

Figure 10 and Figure 11 show the performance of our proposed Reputation Oriented Cluster Bridging strategy (denoted as *SmartQ+*), we compare it with *Flooding*, *Rank* and basic *SmartQ* strategies. When a question cannot receive any question from the asker's social cluster after two forwarding attempts, *Flooding* strategy forwards the question to all other social clusters; *Rank* strategy forward the question to the cluster which has the highest average best answer rate among all its users; and basic *SmartQ* abandons the questions. Figure 10 shows the question response rate when there are different numbers of new questions posted in the system every 30 minutes. It shows that the basic *SmartQ* achieves a question response rate of 0.87. Our proposed *SmartQ+* is effective in further increasing the question response rate, due to the reason that the question is forwarded to the social cluster which has the highest reputation in the question's specific area. *Rank* also improves the response rate as a question is forwarded to another cluster with high average best answer rate, thus increasing the chance of forwarding the question to possible answerers. When question post rate is low, *Flooding* can effectively increase question response rate as the question is flooded to all clusters. However, when the question post rate is high, *Flooding* reduces the question response rate, as questions are flooded to all cluster and users' answering capacities are easily exhausted by the incoming questions. Figure 11 shows the average answer quality provided by other clusters when there are different numbers of new questions posted in the system. As we are only interested in the answers from other cluster, and basic *SmartQ* does not forward unsolved questions to other cluster, so we regard the answer quality as 0. *Flooding* and *Rank* do not consider the capability and willingness of users to answer questions while forwarding questions to other clusters. *SmartQ+* reaches an average answer quality of 0.5, which indicates that the cluster bridging strategy is advantageous in improving the answer quality.

Figure 12 and Figure 13 show the performance of our proposed Lightweight Spammer Detection strategy (denoted by *Spammer+*). We compare it with *Spammer* strategy, in *Spammer*, each user has a 20% probability of reporting a spammer to the system, and 5% probability of falsely reporting a normal user as a spammer. Figure 12 shows the precision rate of different spammer detection strategy when there are different percentages of spammers in the system. We see that *Spammer+* outperforms *Spammer* in increasing the precision rate of spammer detection. Figure 13 shows the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2735442, IEEE Transactions on Big Data

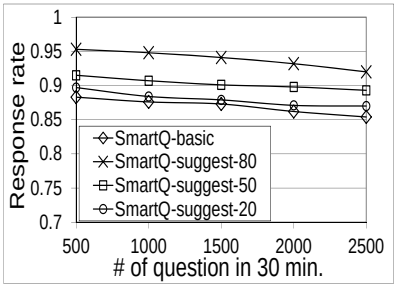


Fig. 14: The question response rate.

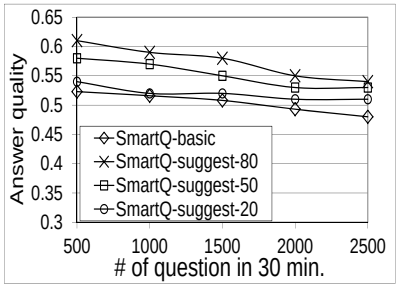


Fig. 15: The average quality of answers.

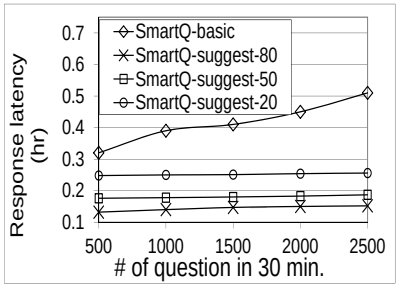


Fig. 16: Average latency of receiving answers.

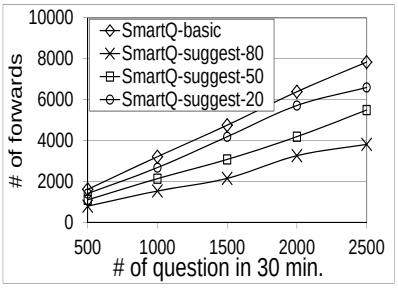


Fig. 17: # of forwards.

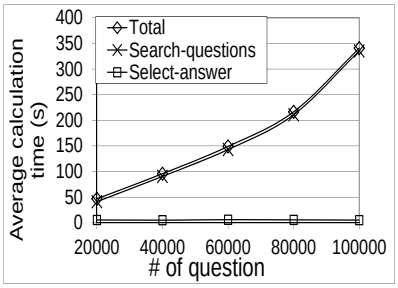


Fig. 18: Calculation time of suggesting answers for new questions.

false positive rate of different spammer detection strategy when there are different percentage of spammers in the system. We see that *Spammer+* exhibits a low false positive rate.

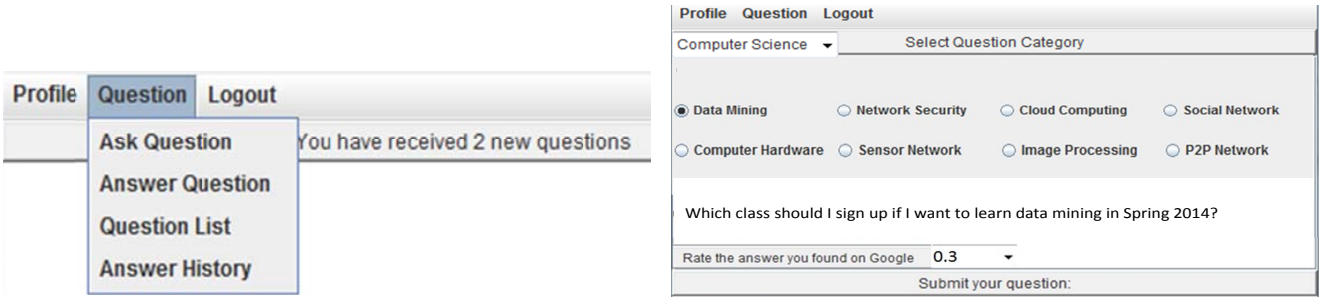
We implemented *SmartQ* with the strategy of suggesting answers to reduce the loads on experts introduced in Section 3.7, which is denoted by *SmartQ-suggest*. We compared *SmartQ-suggest* with the basic *SmartQ* system (denoted by *SmartQ-basic*) in terms of different performance metrics. In *SmartQ-suggest*, we identified top 3 similar questions for each newly generated question, i.e., w in Section 3.7 is set to 3. We then suggested the best answers or high-quality answers of these similar questions to the asker. Each asker decides whether to accept the suggested answers based on his/her subjective judgement. The new question will not be forwarded to other users if the asker accepts the suggested answers. In this experiment, we assumed that each asker has x probability to accept the suggested answers. We developed three variances of *SmartQ-suggest* by setting different values of x, denoted by *SmartQ-suggest-x*. For example, *SmartQ-suggest-50* denotes the system in which each asker has 30% probability to accept the suggested answers.

If an asker accepts the answers suggested by *SmartQ-suggest* for his/her new question, we regard that this question is responded. Figure 14 shows the question response rate when there are different numbers of new questions posted in the system within 30 minutes. We see that the response rate drops gradually when the question arrival rate increases due to the reason explained in Figure 5. *SmartQ-suggest* generates higher response rate than *SmartQ-basic* because *SmartQ-suggest* provides suggested answers from similar questions, and new questions can be solved without being forwarded to the experts. Also, when fewer questions are forwarded to the experts, they are more likely to have enough capacity to answer all incoming questions, leading to a higher question response rate. *SmartQ-suggest-80* yields the highest response rate while *SmartQ-suggest-20* generates the smallest response rate. This is because that a question is responded if the asker accepts the answers suggested by *SmartQ-suggest*. When askers have a higher probability to accept the suggested answers, the question response rate is increased. Figure 14 shows that *SmartQ-suggest* is effective in providing a high question response rate by reducing the loads of experts.

Figure 15 shows the average answer quality when new questions are posted in the system at different rates. We see that *SmartQ-suggest* is effective in providing higher answer quality than *SmartQ-basic*, this is because that *SmartQ-suggest* recommends high-quality answers from similar questions to the askers, and the suggested answers are either best answers selected by the askers of similar questions or high-quality answers evaluated by our system. Thus, askers are more likely to receive high-quality answers. *SmartQ-suggest-80* generates a higher answer quality than *SmartQ-suggest-50* and *SmartQ-suggest-20* because askers have a higher probability to accept the suggested answers from similar questions. Also, fewer questions are forwarded to the experts in *SmartQ-suggest-80*, so experts have enough capacity to answer all incoming questions.

Figure 16 shows the average latency of receiving answers when new questions are generated at different rates. We see that *SmartQ-suggest* outperforms *SmartQ-basic* in reducing answering latency. *SmartQ-suggest* aims to reduce the loads of experts by providing suggested answers from similar questions to new questions, and askers can receive answers quickly without forwarding the questions to experts.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2735442, IEEE Transactions on Big Data



(a) Snapshot of *SmartQ*'s main manu.

(b) Snapshot of *SmartQ*'s question page.

Fig. 19: Snapshot of *SmartQ* Q&A system.

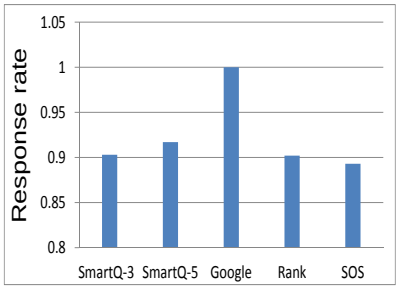


Fig. 20: The question response rate.

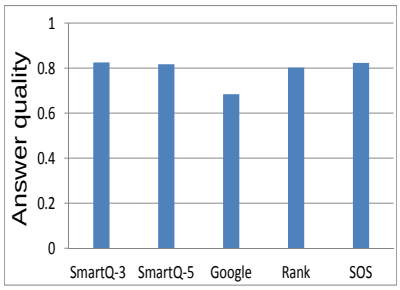


Fig. 21: The average quality of answers.

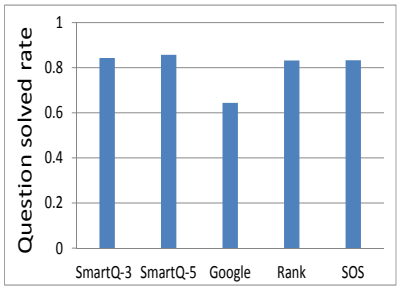


Fig. 22: Question solved rate.

SmartQ-suggest-80 generates a lower latency than *SmartQ-suggest-50* and *SmartQ-suggest-20* because askers have a higher probability to accept the suggested answers. Thus, suggesting high-quality answers from similar questions is crucial to increase the askers' likelihood of accepting the answers and reducing the latency of receiving answers.

Figure 17 shows the total number of question forwarding actions executed with different new questions arrival rates. We see that *SmartQ-suggest* substantially reduces the number of forward actions comparing to *SmartQ-basic*. The relative performance between different variances of *SmartQ-suggest* follows: *SmartQ-suggest-80*<*SmartQ-suggest-50*<*SmartQ-suggest-20* due to the reason that a higher probability to accept the suggested answers leads to fewer forward actions.

Figure 18 shows the average computation time of suggesting answers for new questions when there are different numbers of questions stored in the system. For each new question, we recorded the time needed to search top three similar questions (denoted by *Search-questions*) and the time needed to select answers with high-quality for these questions (denoted by *Select-answer*). We also recorded the total computation time by adding up *Search-questions* and *Select-answer*. We see that the computation time increases gradually as the number of questions stored in the system increases. However, the computation time is generally short compared to the latency of receiving answers from experts in Figure 7.

4.2 Application Implementation and Testing

We developed *SmartQ* client based on Java Applet framework, and the server runs on Tomcat 7.0 using JDBC connector with MySQL. The client is running on any browser supporting Java runtime environment 1.7. 42 students from Clemson University installed *SmartQ* clients and participated in the test. Figure 19 shows the main manu and question page of *SmartQ*. As shown in Figure 19(a), users can ask and answer questions that meet their interests, and check question history. When a user wants to ask a question, he/she is required to select the question category and detailed themes for the question. As shown in Figure 19(b), "Computer Science" is selected as the question category and "Data Mining" as the question theme. Users are also required to search the question on Google and rate the Google results with a score ranged from 0 to 1.

After receiving an answer, an asker needs to rate each answer with a 0-1 score based on the answer quality. The test lasted one month and more than 300 questions were collected and analyzed. The questions were mainly focused on the "Computer Science" category, and multiple themes under this category are presented to further identify the questions. We proposed two different question forwarding strategies: *SmartQ-3* and *SmartQ-5*, which forward a question to 3 and 5 contacts with the highest reputation scores, respectively. We then compared the performance of *SmartQ* with *Google*, *SOS* and *Rank*.

We first examine if users can receive answers for their questions. Figure 20 shows the comparison results of different methods in question response rate. We see that *SmartQ* outperforms *SOS* and *Rank* in drawing answers, due to the reason that the questions are forwarded to users with high reputation, which reflects the users' willingness and ability to answer each specific category of questions. Also, *SmartQ-5* achieves higher question response rate than *SmartQ-3*, because forwarding a question to larger number of users will result in higher chance of reaching a potential answerer, thus

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2735442, IEEE Transactions on Big Data

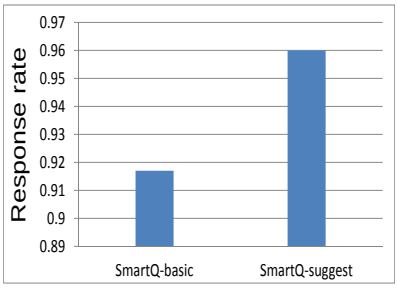


Fig. 23: The question response rate.

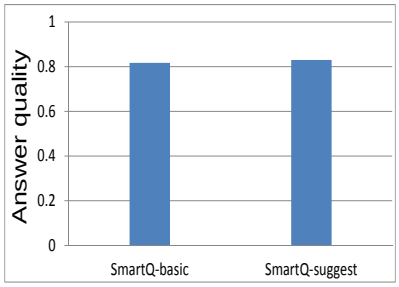


Fig. 24: The average quality of answers.

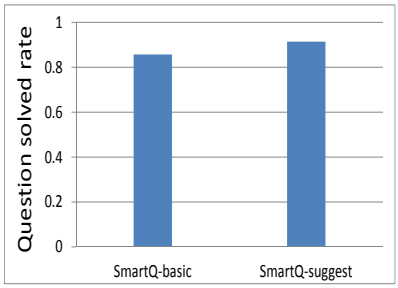


Fig. 25: Question solved rate.

questions are more likely to be solved. *SOS* yields better response rate than *Rank*, as *SOS* aims to match the question's category with potential askers' interests. We assume that *Google* reaches a response rate of 100 percentage as the search engine is always available for information discovery.

Each asker in the test evaluates the answers for his/her questions by assigning quality scores. Figure 21 shows the average answer quality for different strategies. We see that *Google* provides lower quality answers for users than other four strategies, due to the fact that most questions asked by participants in the testing group are non-factual questions. Questions such as "What kind of personal information is safe to disclose on social network?", "What mathematical knowledge is important when studying data mining?", cannot be easily found on Google, but can be solved by users with expertise in that areas. *SOS* considers user interests and expertise when forwarding question, so it gets higher answer quality than *Rank*, which only considers users' ranks they achieve in the system, but does not identify users' expertise and willingness when forwarding new questions. *SmartQ* achieves the highest average answer quality due to the same reason in Figure 6.

When an asker receives a number of answers to his/her question, it is important to determine whether the question is solved or not. In the test, if an asker considers an answer solves the question, he/she will give higher than 0.8 point to this question. Thus, a question is solved if at least one of its answers receives more than 0.8 point. Figure 22 shows the percentage of questions solved by users in different strategies. We see that the question solved rate follows: *SmartQ-5*>*SmartQ-3*>*SOS*>*Rank*>*Google*. *SmartQ* selects potential answerers based on their expertise in the question's area and willingness to answer questions, thus questions are more likely to be solved. *SOS* forwards questions to users who are interested in the questions' area but may not have the ability to answer questions. Thus *SOS* outperforms *Rank*, which does not match potential answerers' expertise to the questions' area. For most non-factual and subjective question, *Google* is incapable of providing satisfying answers.

In the test, we chose 4 random students to be spammers, who were responsible of answering questions with advertisements or randomly generated words. After receiving a spam to his/her question, the asker can report it to the system by clicking the "Report Spam" button, or choose to ignore the spam without reporting. We tested two different methods for spammer detection, one is the proposed lightweight detection strategy and the other is *Reported-based*. In *Reported-based*, if a user is reported as potential spammer by at least 3 users, the system will finally regard he/she as a spammer. Both two strategies are able to identify all 4 spammers, which indicates the effectiveness of our proposed lightweight spammer detection strategy.

Figure 23 shows the question response rate for different strategies. We see that *SmartQ-suggest* outperforms *SmartQ-basic* in providing users with a high response rate. This is due to the same reason explained in Figure 14.

Figure 24 shows the average answer quality for different strategies. We see that *SmartQ-suggest* generates answers with a higher quality than *SmartQ-basic*, due to the fact that *SmartQ-suggest* recommends high-quality answers to askers from similar questions as explained in Figure 15.

Figure 25 shows the percentage of questions solved by users. We see that *SmartQ-suggest* generates answers with higher qualities than *SmartQ-basic* because *SmartQ-suggest* recommends high-quality answers to askers and a new question is likely to be solved by these suggested answers.

5 Conclusions

The rapid growth of Q&A systems make they important ways of knowledge discovery. However, as Q&A systems are generally serving a large amount of users and tens of thousand of new questions are posted in the system everyday, forwarding questions to experts who are willing and able to provide satisfying answers is crucial in maintaining the performance of Q&A systems. Also, in order to improve user loyalty and experience, Q&A systems should be able to identify users who intentionally spread useless information or post advertisements. This paper proposes SmartQ, a reputation based Q&A System. SmartQ evaluates users' reputation towards every knowledge category and theme, and forwards questions to a number of reputable users in the question's knowledge category and theme. SmartQ incorporates a lightweight spammer detection strategy, which examines a user's best answer rate and number of contacts. Also, we proposed a strategy to recommend suggested answers for similar questions to each new question, which can reduce the number of questions forwarded to the experts to reduce their loads. The advantage of SmartQ is verified by experiments on PeerSim and real application. In our future work, we will study using effective incentives to further improve answer quality and response rate, and detecting malicious users by analyzing their behaviors.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TBDATA.2017.2735442, IEEE Transactions on Big Data

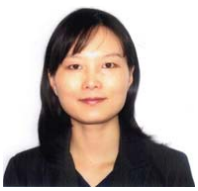
ACKNOWLEDGEMENTS

This research was supported in part by U.S. NSF grants NSF-1404981, IIS-1354123, CNS-1254006, CNS-1249603, Microsoft Research Faculty Fellowship 8300751. We would like to acknowledge Dr. Shui Yu from Deakin University for his valuable discussions and comments. An early version of this work was presented in the Proceedings of ICPADS 2014 [40].

REFERENCES

[1] Yahoo!Answers. http://answers.yahoo.com/, [Accessed in March 2014].
[2] Z. Li, H. Shen, and J. Grant. Collective intelligence in the online social network of yahoo!answers and its implications. In *Proc. of CIKM*, 2012.
[3] H. Shen, Z. Li, J. Liu, and J. Grant. Knowledge Sharing in the Online Social Network of Yahoo! Answers and Its Implications. *TC*, 64(6):1715–1728, 2015.
[4] U. Lee, H. Kang, E. Yi, M. Yi, and J. Kantola. Understanding Mobile Q&A Usage: An Exploratory Study. In *Proc. of CHI*, 2012.
[5] R. White, M. Richardson, and Y. Liu. Effects of community size and contact rate in synchronous social Q&A. In *Proc. of SIGCHI*, 2011.
[6] G. Hsieh, R. Kraut, and S. Hudson. Why Pay?: Exploring How Financial Incentives are Used for Question & Answer. In *Proc. of CHI*, 2010.
[7] K. Nam, M. Ackerman, and L. Adamic. Questions in, knowledge in?: a study of naver's question answering community. In *Proc. of CHI*, 2009.
[8] F. Harper, D. Raban, S. Rafaeli, and J. Konstan. Predictors of answer quality in online Q&A sites. In *Proc. of CHI*, 2008.
[9] L. Gomes, C. Cazita, J. Almeida, V. Almeida, and W. Meira. Characterizing a spam traffic. In *Proc. of IMC*, 2004.
[10] A. Ramachandran and N. Feamster. Understanding the network-level behavior of spammers. In *Proc. of SIGCOMM*, 2004.
[11] E. Tan, L. Guo, S. Chen, X. Zhang, and Y. Zhao. Spam behavior analysis and detection in user generated content on social networks. In *Proc. of ICDCS*, 2012.
[12] Google Answers. http://answers.google.com/, [Accessed in March 2014].
[13] StackOverFlow. www.StackOverFlow.com/, [Accessed in March 2014].
[14] L. Adamicand J. Zhang, E. Bakshy, and M. Ackerman. Knowledge Sharing and Yahoo Answers: Everyone Knows Something. In *Proc. of WWW*, 2008.
[15] F. Harper, D. Moy, and J. Konstan. Facts or Friends? Distinguishing Informational and Conversational Questions in Social Q&A Sites. In *Proc. of CHI*, 2009.
[16] D. Horowitz and S. D. Kamvar. The anatomy of a large-scale social search engine. In *Proc. of WWW*, 2010.
[17] M. Richardson and R. W. White. Supporting synchronous social q&a throughout the question lifecycle. In *Proc. of WWW*, 2011.
[18] S. J. H. Yang and I. Y. L. Chen. A social network-based system for supporting interactive collaboration in knowledge sharing over peer-to-peer network. *IJHCS*, 2008.
[19] M. S. Ackerman. Augmenting organizational memory: a field study of answer garden. *ACM TOIS*, 1998.
[20] H. Kautz, B. Selman, and M. Shah. ReferralWeb: Combining Social Networks and Collaborative Filtering. *Communications of the ACM*, 40(3), March 1997.
[21] D. McDonald and M. Ackerman. Expertise Recommender: A flexible recommendation system and architecture. In *Proc. of CSCW*, 2000.
[22] C. Shah, J. Oh, and S. Oh. Exploring Characteristics and Effects of User Participation in Online Social Q&A Sites. *First Monday*, 13(9), 2008.
[23] U. Lee, J. Kim, E. Yi, J. Sung, and M. Gerla. Analyzing Crowd Workers in Mobile Pay-for-Answer Q&A. In *Proc. of CHI*, 2013.
[24] J. Guo, S. Xu, S. Bao, and Y. Yu. Tapping on the Potential of Q&A Community by Recommending Answer Providers. In *Proc. of CIKM*, 2008.
[25] Y. R. Tausczik and J. W. Pennebaker. Predicting the Perceived Quality of Online Mathematics Contributions from Users' Reputations. In *Proc. of SIGCHI*, 2011.
[26] K. Pelechrinis, V. Zadorozhny, and V. Oleshchuk. Collaborative assessment of information provider's reliability and expertise using subjective logic. In *Proc. of CollaborateCom*, 2011.
[27] P. Long, N. Anh, N. Vi, L. Quoc, and H. Thang. A meaningful model for computing users' importance scores in Q&A systems. In *Proc. of SoICT*, 2011.
[28] R. Delaviz, J. Pouwelse, and D. Epema. Targeted and scalable information dissemination in a distributed reputation mechanism. In *Proc. of STC*, 2012.
[29] M. Blaze, J. Feigenbaum, and A. Keromytis. Keynote: Trust management for public-key infrastructures (position paper). In *Proc. of Security Protocols Workshop*, 1998.
[30] X. Si, E. Chang, Z. Gyongyi, and M. Sun. Confucius and its intelligent disciples: integrating social with search. *Proceedings of the VLDB Endowment*, 3(1-2):1505–1516, 2010.
[31] J. Jeon, W. Croft, and J. Lee. Ncr: A scalable network-based approach to co-ranking in question-and-answer sites. In *Proc. of CIKM*, 2005.
[32] A. Rajaraman and J. Ullman. *Mining of massive datasets*, volume 1. Cambridge University Press Cambridge, 2012.
[33] A. Shtok, G. Dror, Y. Maarek, and I. Szpektor. Learning from the past: answering new questions with past answers. In *Proc. of WWW*, 2012.
[34] J. Jeon, B. Croft, J. Lee, and S. Park. A framework to predict the quality of answers with non-textual features. In *Proc. of ACM SIGIR*, 2006.
[35] B. John, A. Chua, and D. Goh. What makes a high-quality user-generated answer? *Internet Computing*, 15(1):66–71, 2011.
[36] L. Larkey. Automatic essay grading using text categorization techniques. In *Proc. of ACM SIGIR*, 1998.
[37] Peersim: A peer-to-peer simulator. http://peersim.sourceforge.net/, [Accessed in March 2014].
[38] Z. Li, H. Shen, G. Liu, and J. Li. SOS: A Distributed Mobile Q&A System Based on Social Networks. In *Proc. of ICDCS*, 2012.
[39] J. Janes, C. Hill, and A. Rolfe. Ask-an-expert services analysis. *JASIST*, 52(13):1106–1121, 2001.
[40] Y. Lin and H. Shen. Smartq: A question and answer system for supplying high-quality and trustworthy answers. In *Proc. of ICPADS*, 2014.

Yuhua Lin Yuhua Lin received both his BS degree in Software Engineering and MS degree in Computer science from Sun Yat-sen University, China in 2009 and 2012 respectively. He is currently a Ph.D student in the Department of Electrical and Computer Engineering of Clemson University. His research interests focus on effective and economical content delivery strategy on the cloud.



Haiying Shen Haiying Shen received the BS degree in Computer Science and Engineering from Tongji University, China in 2000, and the MS and Ph.D. degrees in Computer Engineering from Wayne State University in 2004 and 2006, respectively. She is currently an Associate Professor in the Department of Electrical and Computer Engineering at Clemson University. Her research interests include distributed computer systems and computer networks, with an emphasis on P2P and content delivery networks, mobile computing, wireless sensor networks, and grid and cloud computing. She was the Program Co-Chair for a number of international conferences and member of the Program Committees of many leading conferences. She is a Microsoft Faculty Fellow of 2010 and a member of the IEEE and ACM.